Fable II: Design of a Modular Robot for Creative Learning

Pacheco, Moises; Fogh, Rune; Lund, Henrik Hautop ; Christensen, David Johan

Published in:
Proceedings of 2015 IEEE International Conference on Robotics and Automation.

Link to article, DOI:
10.1109/ICRA.2015.7140060

Publication date:
2015

Document Version
Peer reviewed version

Fable II: Design of a Modular Robot for Creative Learning

Moises Pacheco, Rune Fogh, Henrik Hautop Lund and David Johan Christensen

***Abstract*— Robotic systems have a high potential for creative learning if they are flexible, accessible and engaging for the user in the experimental process of building and programming robots. In this paper we describe the Fable modular robotic system for creative learning which we develop to enable and motivate anyone to build and program their own robots. The Fable system consists of self-contained modules equipped with sensors and actuators, which users can use to easily assemble a wide range of robots in a matter of seconds. The robots are user-programmable on several levels of abstraction ranging from a simple visual programming language to powerful conventional ones. This paper provides an overview of the design of Fable for different user groups and an evaluation of critical issues when we attempt to integrate the system into an everyday teaching context.**

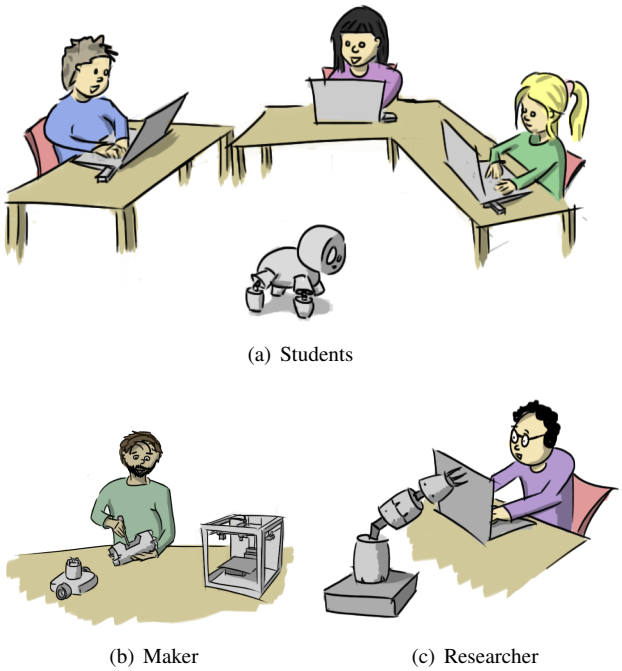
I. INTRODUCTION

Today's world is filled with consumer products that constantly encourage us to buy and not to build. Taught to us from an early age, plagiarism and copyright policies serve as mental barricades that dry out our curiosity, creativity and collaboration [1]. In this work we seek to revitalize and quench our users thirst for knowledge within the domain of robotics. We believe that, given the right tools, anyone can become a robot designer.

In this paper[1] we present the design of Fable, a mechatronic construction kit that allows users to playfully build and program their own robots. Our objective is to motivate users both towards making their own robots and sharing them with others, that is as a DIY (Do it Yourself) kit and as a DIT (Do it Together) kit.

We have designed Fable as a modular robotic platform with a focus on the users' needs, ranging from a classroom of kids, and after-school clubs, to hobbyists/makers and even researchers, as illustrated in Fig 1. This diversity in usability is achieved by encapsulating key robotic functionalities into modules that can be combined in numerous configurations utilizing a shared connector and communication system. This gives us the freedom to design basic modules for kids and high-end modules for researchers while making it easy for makers to start building their own. To support this diversity of users we enable them to program the system by using their preferred programming language (Blockly, Python and Java are currently supported and we plan support for Matlab).

The rest of this paper starts by describing related work (Sec. II). It continues by presenting the design of Fable, that is: mechanics, electronics and software (Sec. III). Further in Sec. IV the paper exemplifies how robots can be assembled in seconds, programmed with Blockly and Python, and we evaluate Fable as an educational platform based on programming sessions with users and experts.



(a) Students

(b) Maker

(c) Researcher

Fig. 1. Prospective Fable users with different interests, objectives and level of experience.

II. RELATED WORK

The Fable system is designed to support the user's creative thinking and innovation. In order to guide the development of creative toolkits for users, Resnick et al. has proposed a set of design principles such as "Low Threshold, High Ceiling, and Wide Walls" [3]. Von Hippel described toolkits for user innovation as a way to transfer design abilities from the toolkit developer to the user [4]. Further, Von Hippel proposed five key objectives for such toolkits: 1) Enable the user to perform trial-and-error learning, 2) span a solution-space, that embraces what the user wants to design, 3) is user-friendly by being familiar and easily accessible to the user, 4) contains a standard library, that users can combine with their own designs, 5) automatically translate the user's design into the format required to produce the design. In this work, we are guided by such design principles and objectives in order to make Fable as valuable as possible for its users.

Fable is a modular robotic system. Such systems achieve flexibility and versatility through modularity and thereby

Technical University of Denmark (DTU), Center for Playware, Department of Electrical Engineering, Elektrovej, Building 326, DK-2800 Kgs. Lyngby, Denmark {mpac| djchr| hhl} @elektro.dtu.dk, rufo@dtu.dk

[1]An earlier version of this paper was presented at the IROS 2014 workshop on Modular and Swarm Systems [2]

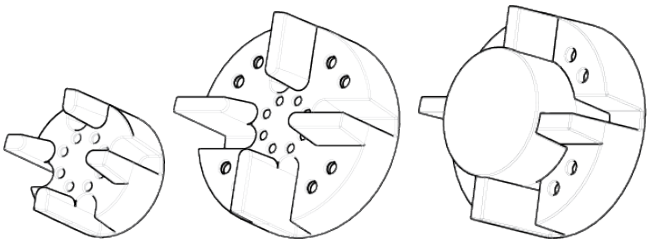
provides users with what is known as "Wide Walls". Modular robotic systems consist of a collection of simple robotic units that can attach and detach from each other to form a wide range of configurations [5], [6]. While the majority of modular robots are designed to study self-reconfiguration, e.g. [7], [8], [9], Fable takes its inspiration mainly from user reconfigured and interactive systems. Fable is more similar to Roblocks/cubelets [10], MOSS [11], Topobo [12] and LEGO Mindstorms in that all these systems aims to enable and motivate everyone to become a robot designer.

With Fable we aim to provide a creative experience to its users. Dahl and Moreau defined experiential creation as *"...activities in which a consumer actively produces an outcome"* [13]. For robots this definition ranges from kits which the user assembles into a specific predefined robot to open-ended systems where little or no guidance is given to the user in how the robot should be designed. The modularity and granularity of Fable provides constraints, and thereby guidance, to the solution-space but the system is prepared for makers to overcome such constrains, e.g. by creating their own types of modules.

Ideally, Fable should motivate the users to be creative and learn in the process. As observed by Dahl and Moreau [13] users are motivated to engage in creative work for both intrinsic and extrinsic reasons, including a feeling of accomplishment, a desire to learn and to share a creative experience with others. Further, users are motivated by the satisfaction they feel from an immersion in the creative process [13]. This immersion is related to the mental state of flow which is characterized by a lost sense of time and being fully absorbed in the current activity [14]. A state of flow is more likely when the user feels that the activity i) has a clear goal, ii) has immediate feedback on the users performance and iii) has an appropriate balance between the challenges of the task and the user's perception of own skills [14]. Play share many characteristics with flow and, as argued by Brown, is ubiquitous in nature and critical for both children and adults development and well being [15]. Play can motivate users to perform learning activities and is therefore a key objective when designing interactive learning environments [16]. The PlayGrid is a model which aims to encapsulate what makes a user enter a state of Play, based on four types of play: the Assembler, the Director, the Explorer, and the Improviser [17].

Educational robotic kits such as Lego Mindstorms and VEX are widely used in classrooms for their mechanical flexibility and their easy to use programming environment. These kits, including Fable are inspired by the learning theory of constructionism [18] and aim to move users away from the passive individual thinking and into active hands-on collaborative learning-by-building. An important difference between Fable and fine granularity systems, such as Lego Mindstorms, is the reduced effort required by the user to modify and experiment with their mechanical design.

A visual programming language enables a "Low Threashold" entry to robot programming. For Fable we utilize Blockly [19], a block programming approach similar to Scratch [20], to allow non expert users to graphically program and interact with the system. In order to address the needs of different users, to provide scaffolding and a "High Ceiling" also Python, Java and eventually Matlab can be used to program the system.



(a) Small size (b) Medium size (c) Compatibility of two connector sizes

Fig. 2. Connector design: With the current design any size is possible and compatible with the rest, having only as a lower limit the small size connector diameter.

III. Fable Design

This section provides an overview of the design of our second version of the Fable system, details about the first version can be found in our previous work [21], [22], [23].

A. Mechanics

Our approach uses powerful, yet easy to connect modules that allow users to assemble a functional robot in a matter of seconds. The Fable system is divided in active and passive modules. Active modules contain a set of electronic boards with a microcontroller, onboard power, and a radio device for wireless communication with a PC. These modules also provide functionalities through actuation and sensing, e.g. one active module design is a 2 degree of freedom joint, see Fig. 4(a). Passive modules consist of a variety of shapes made out of empty plastic shells. These passive modules help give the robot structure and shape, e.g. a 'Y' shaped module is used to connect three modules together and an 'X' to connect four. Both can be seen in the robot configuration in Fig. 4(b).

A key feature in modular systems are the connectors since they serve as the only contact surface between modules. Our current connector design, illustrated in Fig. 2, is genderless and four way redundant, which allows our users to explore several connection possibilities between modules. Each connector has at least one ring of magnets that attaches to a matching set on the connecting end. The connector uses a set of flanges that lock the modules allowing only the user to disconnect them by pulling them apart. With this design we obtain a strong connection between modules and yet it's easy enough for children to disconnect. The connector design is scalable, meaning that it is compatible between different sizes, giving us the possibility of combining large modules with small ones, as illustrated on Fig. 2(c).

B. Electronics

For the Fable system we have developed a set of electronic boards, that when combined with commercially available

boards give us a modular electronic configuration. The electronic boards are designed for simplicity, low production cost, flexibility and hackability. The modular approach enables us to create different active modules by mixing electronics boards in new configurations. Table I describes the electronic boards currently used in Fable. Different active modules and a radio dongle will use a specific subset of the electronics modules. To facilitate hackability the module and dongle firmware is executed on an embedded Arduino board. As we develop new types of Fable modules we will develop new modular electronic boards to support them.

Board Name	Details	Description
Module Board	11.1v, 1000mAh LiPo battery	Main mother board for active modules
Dongle Board	5v USB powered	Main mother board for dongles
User Interface Board	RGB diode, button, buzzer, recharge plug, on/off switch	Functions for user feedback and direct interaction with module/dongle
Motor Board	RS232 half-duplex dual buffer	Interface board for serial control of Dynamixel motors (e.g. AX-12A)
Arduino Pro Mini	AVR Atmega328, 3.3v, 8MHz	MCU module for dongles and modules
Radio Board	NRF24L01+, 2.4GHz, 2MBit, SPI interface	Wireless communication between modules and dongles

TABLE I

OVERVIEW OF ELECTRONICS BOARDS USED IN FABLE

C. System Network Architecture

The underlying objective of the Fable network architecture is to make the robot programming as simple and flexible as possible. Due to a low lag radio communication link to the modules, the user can program the distributed robots as if it was centralized and connected directly to the PC. Therefore, the user avoids the difficulties of cross-compiling, downloading program to robot and debugging a distributed embedded platform.

The network architecture of the Fable system is shown in Fig. 3. The user PC is serially connected to a dongle which provides a shared 2 Mbit radio communication link between the user controlled application and the modules. Modules are addressed using an ID and their module type. Web services can be used to enhance the possibilities of the user application, currently we use a web service API for speech generation. Although the radio communication is shown here as a master-slave architecture, it can also function as a peer-to-peer network. This can be exploited in certain research studies, e.g. on distributed control as described in Section III-D. In addition, we plan to exploit asynchronous communication between different dongles to enable the different users to program collaboratively by writing different parts of a program and remotely calling functions developed by other users.

D. Distributed Hardware-in-the-loop Control

Research within modular and swarm robotics often explores distributed control strategies, e.g. for behaviors such as locomotion, learning and shape-formation [24], [25], [26]. Debugging and validating distributed controllers is challenging and hardware-in-the-loop simulators can facilitate the development process [27].

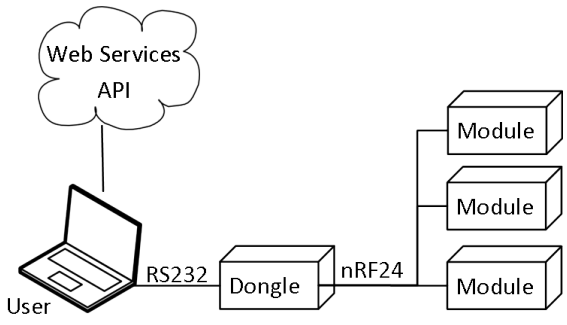


Fig. 3. Network architecture of the Fable system.

Fable is mainly centralized controlled given that it receives commands wirelessly from a PC and lacks dedicated neighbor-to-neighbor communication. However, Fable can also be used as a hardware-in-the-loop simulator for developing and validating distributed control strategies. The software API supports distributed control where each module controller executes in a separate thread on the PC. Also, passive modules will run independently in a control thread to take part in the communication network. Given an adjacency matrix of the robot the modules can simulate neighbor-to-neighbor communication. Fable supports two modes of distributed communication:

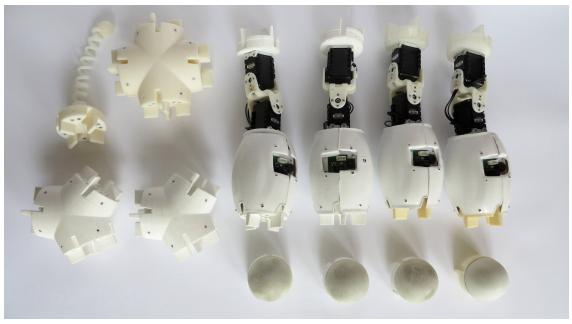
Simulated: Messages are exchanged between module control threads locally on the PC. Transmissions are performed without significant lag, without payload length limits and messages are never lost.

Hardware-In-The-Loop: Messages are wirelessly transmitted between two real-world modules for a more realistic performance. The lag of transferring such a message varies from approx. 6 ms to 13 ms with the length of the payload (0 to 26 bytes respectively). Message loss heavily depends on the specific setup, for example in one setup with 15 meters between two modules we measured package loss to increase from $4.6 \pm 2.7\%$ to $43.4 \pm 22.6\%$ when a human moved within the line-of-sight between the modules.

E. User Programming

The user develops the system application from a personal computer using one of the supported programming languages. When the user executes the application program it is not cross-compiled but run locally on the PC. Simple APIs enable the user-program to call functions on the remote modules through the dongle connected to the PC (based on module IDs). The round-trip lag for a simple remote procedure call is around 4.2 ms, longer if the remote module needs to perform processing. This is sufficiently low for most applications and can be reduced even further in future work.

We developed a Graphical User Interface (GUI), based on Blockly [19], as a means for non-programmers to quickly



(a) Modules

(b) Quadruped

Fig. 4. (a) Four active joint modules and eight passive modules. (b) A Fable quadruped robot assembled from the above twelve modules.

start developing applications. Blockly is an open-source framework for building application specific visual programming languages inspired by the Scratch programming environment [20]. An example of a Blockly program for controlling a simple interactive Fable robot is shown in Fig. 5(a). The source output generated from Blockly is written in Python. This string of Python code is send from the Blockly JavaScript web-application through an HTTP request to a Python server which executes the Python control application. Further, the Blockly interface enables users to collect and analyze data from the robot, e.g. as real-time-plots of sensor values.

More experienced users can also write applications directly in either Python or Java based on APIs. Further, we plan to add Matlab support for research and university education in future work.

IV. Examples and Evaluation

This section describes an experiment to measure the assembly time of a robot, present an example of Blockly programming and present qualitative observations based on programming sessions with Fable.

A. Assembly and Disassembly Speed

This example illustrates the assembly of a typical robot that could be built and programmed by most non-expert users. In programming sessions students have previously programmed similar quadruped with different gait patterns, and in the process learned about programming, robotics, and mathematics. In such sessions, we have observed the importance of fast and easy reconfiguration which motivates the students to explore and experiment with the robot morphology.

Fig. 4(a) shows the modules as an exploded view of the Quadruped shown in Fig 4(b). The quadruped consist of eight passive modules: four feet, two 'Y', one 'X' and one tail module. In addition, four active joint 2DOF modules are used in the quadruped.

We have measured that it takes on average 16.9 seconds to assemble and 9.8 seconds to disassemble the robot for an experienced user. This corresponds to 1.5 second for connecting two modules and 0.8 seconds to disconnect. For comparison, Davey et al. reported on the assembly time of CKBot with the ModLock connector which took 7 seconds per connection and 3 seconds per disconnect in a snake configuration [28]. Compared to many other systems, both Fable and CKBot are fast to assemble, e.g. it takes 1917 seconds to assemble a simple Bioloid snake, as reported by Davey et al. [28].

B. Visual User-Programming

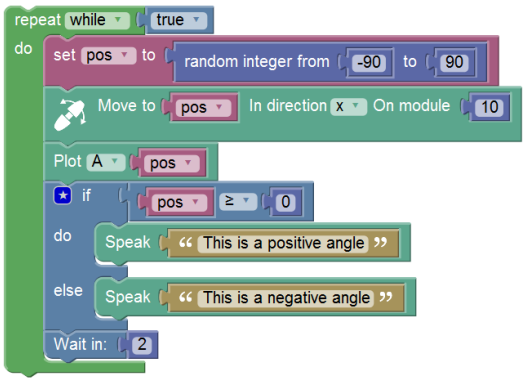
A simple example of a Blockly program, which can be used to control Fable, is shown in Fig. 5(a). This program controls a joint module to random angles and reports to the users which angles are positive and negative while illustrating the angles on a plot. Students can build such a program as an exercise to help them visualize angles in different contexts: plots, motors, degrees.

The program runs in an infinite loop where it starts by assigning a random integer to a variable. The variable is then used as an angle set-point on the joint module with ID number 10. Then it continues to plot the motor's angular position and evaluates if the position is positive or negative and speaks its correspondent sentence. The speak function uses a web API to speak a sentence, with local playback on the PC. Further, the plot function plots the position in real-time on a graph which is an integrated part of the graphical user interface. Plotting is useful for debugging and educational purposes.

In Fig. 5(b) the equivalent Python source code generated from the Blockly program is shown. We anticipate that the similarity between Blockly and Python will ease beginner programmers' to transition from a visual programming language to a general purpose programming language. More advanced control strategies e.g. for research experiments could be implemented in Python, Java and in the near future also in Matlab.

C. Programming Sessions

In order to evaluate Fable as an educational platform we arranged a number of programming and building sessions with educational experts, schoolteachers, and students from different age groups (ages 8 and up). Some of the sessions were combined with semi-structured interviews. The focus



(a) Blockly

```python
api.startup()
import random
pos = None
while True:
    pos = random.randint(-90, 90)
    api.setModuleMotorPosition(10, 0, pos)
    api.plotDataAppend(pos, label ='A', windowSize = 50 )
    if pos >= 0:
        api.speak(str('This is a positive angle'))
    else:
        api.speak(str('This is a negative angle'))
    api.sleep(2)
api.sleep(1)
api.terminate()
```

(b) Python

Fig. 5. (a) A simple example Blockly program for controlling a single Fable joint module (ID=10). (b) The corresponding Python code output by Blockly.

was on late primary and secondary school in order to learn how the system could be integrated in a school context. For practical reasons we focused our attention on the Danish school system but we anticipate that our observations may generalize to other school systems as well. Based on these sessions we identified critical themes that will guide the further development of Fable:

a) Classroom integration: The structure of everyday teaching in schools affects how robotics can be integrated in the classroom. In the Danish school system the classes are organized in 45 minute lessons. In a typical lesson the teacher first lectures the topic followed by students working on exercises related to the topic. In order for robotics to fit in this time-scarce context, it is important that the setup time of the system is kept to a minimum. However, most robotic kits require more time in the assembly phase, e.g. the LEGO Mindstorms Education EV3. A minimal EV3 mobile robot requires the user to follow a 40-step assembly manual (equivivalent to estimated 20-40 min.) [29]. Mechanical assembly arguably has valuable learning outcomes, but in practice it limits the use of robots to longer integrated projects or the use of pre-assembled fixed morphology robots. In order to address this issue, in the design of Fable our objective is to keep the setup and assembly time as low as possible.

b) Educational material alignment: Creating and programming robots freely based on intrinsic motivation can be a highly rewarding and educational experience for the students. However, in order to achieve acceptance within the conventional school system educational material must be designed in order to realize specific learning outcomes. In the Danish school system such learning objectives are defined at a national level, but how to teach them is left up to the individual schools. Most schools base their teaching on a standard text books which are written for a specific grade covering one class. For Fable the most relevant classes are Math and Physics, and to a lesser extent Biology and English. Therefore, in order for Fable to fit in a classroom context educational material must be developed that is closely aligned with the national learning objectives. We anticipate that such educational material will most likely lead to the development of new Fable module types.

c) Motivating teachers: Robotic toolkits have the potential to provide engaging hands-on learning experience to students. However, as noted by Bers et al., few teachers have the experience and skills necessary to integrate such toolkits in the classroom [30]. This stresses the importance of designing an easily accessible robot system, training the teacher in their usage and providing them with familiar materials that will make the learning process for them both enjoyable and rewarding. It is a primary concern of the Fable system to motivate the teachers and give them the necessary confidence to use the system in the classroom.

d) Motivating students: Students enjoyment is affected by how the learning situation is structured. In one extreme the students follow a strict tutorial to build and program a specific robot. At the other extreme the students are left without any constraints to build whatever they want. Dahl and Moreau found that providing users with instructions and a general goal, but no specific target, would result on a higher feeling of competence, autonomy and task enjoyment compared to situations with no instruction or a specific target [13]. In the context of Fable, we have observed that a lack of instruction and clear goals are likely to quickly make the students frustrated. On the other hand, we have also observed how good instruction combined with clear goals such as challenges, competitions and performances made the students highly engaged in the activity. How to integrate such motivating activities in a time-scarce context with well defined learning objectives is a critical topic for further research.

V. Conclusion

In this paper we described the design of the second version of Fable. Fable is a modular robotic platform that enables non-technical users, ranging from young students, makers and researchers to assemble and program their own interactive robots. We described the mechanics of the system with passive and active modules and a scalable and robust

connector design, that enables users to build and reconfigure a robot in a matter of seconds. The paper also provides an overview of our modular electronic design and how this allows us to build and develop new types of modules faster. Further, we described how the user could program the system as if it was a centralized robot at different levels of abstraction ranging from a visual programming language to conventional languages such as Python and Java. We also described how the system can be used as a hardware-in-the-loop simulator for research in distributed control strategies. We evaluated the robot and found that a typical configuration could be assembled or disassembled in less than 20 seconds which is important to motivate users for trial-and-error learning and for classroom integration. Further, we identified critical themes for integrating Fable in a school context based on programming sessions with different user groups. In future work we will continue to optimize the Fable system to meet the requirements of its users. Currently, we are extending the system with more passive and active modules, including a gripper, a head module with various sensors and a rotational base module which can also be used as a wheel. Furthermore we are working on interfacing Fable with Matlab and Simulink to allow researchers to simulate algorithms with hardware in the loop.

VI. Acknowledgments

This work was performed as part of the "Modular Playware Technology" project funded by the Danish Advanced Technology Foundation as well as additional funding from the Dr. Techn. A.N. Neergaard og Hustrus foundation and the Siemens Foundation. Thanks to all the members of Center for Playware for their contributions to the project.

References

[1] P. Goodman. *Compulsory miseducation*. Penguin Harmondsworth, 1971.

[2] M. Pacheco, R. Fogh, H. H. Lund, and D. J. Christensen. Fable: A modular robot for students, makers and researchers. In *Proceedings of the IROS workshop on Modular and Swarm Systems: from Nature to Robotics*, 2014.

[3] M. Resnick, B. Myers, K. Nakakoji, B. Shneiderman, R. Pausch, T. Selker, and M. Eisenberg. Design principles for tools to support creative thinking. In *NSF Workshop on Report on Creativity Support Tools*, 2005.

[4] E. Hippel. User toolkits for innovation. *Journal of product innovation management*, 18(4):247–257, 2001.

[5] M. Yim, WM Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *Robotics & Automation Magazine, IEEE*, 14(1):43–52, 2007.

[6] K. Stoy, D. Brandt, and D. J. Christensen. *Self-reconfigurable robots: an introduction*. MIT Press, 2010.

[7] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata. Distributed self-reconfiguration of M-TRAN III modular robotic system. *International Journal of Robotics Research*, 27(3-4):373–386, 2008.

[8] E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund. Design of the ATRON lattice-based self-reconfigurable robot. *Autonomous Robots*, 21:165–183, 2006.

[9] A. Sproewitz, A. Billard, P. Dillenbourg, and A. J. Ijspeert. Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4259–4264. IEEE, 2009.

[10] E. Schweikardt and M. D. Gross. roblocks: a robotic construction kit for mathematics and science education. In *Proceedings of the 8th international conference on Multimodal interfaces*, pages 72–75. ACM, 2006.

[11] ModularRobotics. Moss robot construction system. `http://www.modrobotics.com/moss/`. Accessed: 2014-06-26.

[12] H.S. Raffle, A. J. Parkes, and H. Ishii. Topobo: a constructive assembly system with kinetic memory. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 647–654. ACM, 2004.

[13] D. W. Dahl and C. P. Moreau. Thinking inside the box: Why consumers enjoy constrained creative experiences. *Journal of Marketing Research*, 44(3):357–369, 2007.

[14] M. Csikszentmihalyi. *Flow: The psychology of optimal performance*. 1990.

[15] S. L. Brown. *Play: How it shapes the brain, opens the imagination, and invigorates the soul*. Penguin, 2009.

[16] L. P. Rieber. Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational technology research and development*, 44(2):43–58, 1996.

[17] R. Fogh and A. Johansen. The play grid. *Proceedings of the 12th International Conference on Interaction Design and Children - IDC '13*, pages 356–359, 2013.

[18] Y. B. Kafai and M. Resnick. *Constructionism in practice: Designing, thinking, and learning in a digital world*. Routledge, 1996.

[19] Google. blockly - a visual programming editor. `https://code.google.com/p/blockly/`. Accessed: 2014-06-26.

[20] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[21] M. Pacheco, M. Moghadam, A. Magnusson, B. Silverman, H. H. Lund, and D. J. Christensen. Fable: Design of a modular robotic playware platform. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 544–550. IEEE, 2013.

[22] A. Magnússon, M. Pacheco, M. Moghadam, H. H. Lund, and D. J. Christensen. Fable: Socially interactive modular robot. In *The Eighteenth International Symposium on Artificial Life and Robotics 2013*, 2013.

[23] B. Heesche, E. MacDonald, R. Fogh, M. Pacheco, and D. J. Christensen. Playful interaction with voice sensing modular robots. In *Social Robotics*, pages 180–189. Springer, 2013.

[24] D. J. Christensen, U. P. Schultz, and K. Stoy. A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurable modular robots. *Robotics and Autonomous Systems*, 61(9):1021–1035, 2013.

[25] WM Shen, P. Will, A. Galstyan, and CM Chuong. Hormone-inspired self-organization and distributed control of robotic swarms. *Autonomous Robots*, 17(1):93–105, 2004.

[26] M. Rubenstein, A. Cornejo, and R. Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[27] R. Lal and R. Fitch. A hardware-in-the-loop simulator for distributed robotics. In *Proceedings of ARAA Australasian Conference on Robotics and Automation (ACRA)*, pages 544–550, 2009.

[28] J. Davey, J. Sastra, M. Piccoli, and M. Yim. Modlock: A manual connector for reconfigurable modular robots. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3217–3222, Oct 2012.

[29] *LEGO Education Mindstorms EV3 Building Guide*, 2014.

[30] M. U. Bers, I. Ponte, C. Juelich, A. Viera, and J. Schenker. Teachers as designers: Integrating robotics in early childhood education. *Information Technology in Childhood Education Annual*, 2002(1):123–145, 2002.